PostgreSQL is YeSQL!
PostgreSQL Major Contributor
Dimitri Fontaine

- Extensions
- Event Triggers
- pgloader
- pginstall
- prefix, preprepare, pgstaging...
Relational Database System

- Data Access Service
- Concurrent Reads and Writes
- Multiple Version Concurrency Control
- "Typed" Protocol
- API, not storage, not serialization
CAUTION

ACID
ACID

- Atomicity
- Consistency
- Isolation
- Durability
Atomicity

- BEGIN; ... COMMIT;
- BEGIN; ... ROLLBACK;
- Includes DDL
- Consistent Backups
- Online Backups
- Physical or Logical
Consistency

- NOT NULL
- CHECK constraints
- CREATE DOMAIN
- Primary Key, Foreign Key
- Unique
- Triggers, Constraint Triggers
Consistency

• Relational Model is Strongly Typed
• Data Type Input Function
  • date/time field value out of range:
    “0000-00-00”
    “0000-03-19”
• Exclusion Constraints
Exclusion Constraints

CREATE TABLE reservation
(
    room text,
    professor text,
    during period,

    EXCLUDE USING gist
    (  room with =,
        during with &&
    );
)
PostgreSQL Data Types

- Integer
- Arbitrary precision numbers, UUID
- Floating point
- Character, Text
- Bytea, bitstring
- Date/Time, Time Zones

- Boolean
- Enum, Arrays, Composite Types, Range Types
- Point, Line Segments, Boxes, Paths, Polygons, Circles
- Inet, CIDR, Macaddr
- JSON, XML
PostgreSQL Extensions

- cube
- hstore
- intarray
- ltree
- pg_trgm
- seg
- ip4r
- prefix_range
- pgmp, tinyint
- sha-1, sha-2, md5
- hyperloglog
- and much more
Isolation

- SET TRANSACTION transaction_mode
- ISOLATION LEVEL
  - serializable
  - repeatable read
  - read committed
Durability

- fsync
- synchronous_commit defaults to on
  (off, local, remote_write, on)
- Per-Transaction Control
synchronous_commit

SET demo.threshold TO 1000;

CREATE OR REPLACE FUNCTION public.syncrep_important_delta()
    RETURNS TRIGGER
    LANGUAGE PLpgSQL
AS
$$
DECLARE
    threshold integer := current_setting('demo.threshold')::int;
    delta integer := NEW.abalance - OLD.abalance;
BEGIN
    IF delta > threshold
    THEN
        SET LOCAL synchronous_commit TO on;
    END IF;
    RETURN NEW;
END;
$$;
NoSQL GET/SET

1 Server with 1 to 64 clients, Client(s) and server on separate host
minimum data size: 1188, max size: 2601, average size: 1874
The Real Power of psql
Structured Query
An interesting factoid: the team that recorded the fewest defensive rebounds in a win was the 1995-96 Toronto Raptors, who beat the Milwaukee Bucks 93-87 on 12/26/1995 despite recording only 14 defensive rebounds.
with stats(game, team, drb, min) as
(
    select ts.game, ts.team, drb,
    min(drb) over ()
    from team_stats ts
    join winners w on w.id = ts.game
        and w.winner = ts.team
)
select game.date::date,
    host.name || ' -- ' || host_score as host,
    guest.name || ' -- ' || guest_score as guest,
    stats.drb as winner_drb
from stats
    join game on game.id = stats.game
    join team host on host.id = game.host
    join team guest on guest.id = game.guest
where drb = min;
<table>
<thead>
<tr>
<th>RECORD</th>
<th>Date</th>
<th>Host</th>
<th>Guest</th>
<th>Winner Drb</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1995-12-26</td>
<td>Toronto Raptors</td>
<td>Milwaukee Bucks</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>1996-02-02</td>
<td>Golden State Warriors</td>
<td>Toronto Raptors</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>1998-03-31</td>
<td>Vancouver Grizzlies</td>
<td>Dallas Mavericks</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>2009-01-14</td>
<td>New York Knicks</td>
<td>Washington Wizards</td>
<td>14</td>
</tr>
</tbody>
</table>
PostgreSQL JOINs

• Nested Loop
• Merge Join
• Hash Join
• Semi Join
• Anti Join
• Inner Join
• Outer Join
• Cross Join
• Lateral Join
Window Functions

```sql
# select x,
array_agg(x) over (order by x)
from generate_series(1, 3) as t(x);
```

<table>
<thead>
<tr>
<th>x</th>
<th>array_agg</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1}</td>
</tr>
<tr>
<td>2</td>
<td>{1,2}</td>
</tr>
<tr>
<td>3</td>
<td>{1,2,3}</td>
</tr>
</tbody>
</table>

(3 rows)
Window Functions

```
# select x,
    array_agg(x) over () as frame,
    sum(x) over () as sum,
    x::float/sum(x) over () as part
from generate_series(1, 3) as t(x);

<table>
<thead>
<tr>
<th>x</th>
<th>frame</th>
<th>sum</th>
<th>part</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1,2,3}</td>
<td>6</td>
<td>0.166666666666667</td>
</tr>
<tr>
<td>2</td>
<td>{1,2,3}</td>
<td>6</td>
<td>0.333333333333333</td>
</tr>
<tr>
<td>3</td>
<td>{1,2,3}</td>
<td>6</td>
<td>0.5</td>
</tr>
</tbody>
</table>
```

(3 rows)
Window Functions

```sql
# select x,
    row_number() over(),
    ntile(4) over w,
    lag(x, 1) over w,
    lead(x, 1) over w
from generate_series(1, 15, 2) as t(x)
window w as (order by x);
```

<table>
<thead>
<tr>
<th>x</th>
<th>row_number</th>
<th>ntile</th>
<th>lag</th>
<th>lead</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>7</td>
<td>4</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>4</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

(8 rows)
Histograms
with drb_stats as (  
  select min(drb) as min,  
      max(drb) as max  
  from team_stats  
),  
  histogram as (  
  select width_bucket(drb, min, max, 9) as bucket,  
      \text{int4range}(min(drb), max(drb), '[]') as range,  
      count(*) as freq  
  from team_stats, drb_stats  
  group by bucket  
  order by bucket  
  )  
  select bucket, range, freq,  
      \text{repeat}('\ast', (freq::float / max(freq) over() \ast 30)::int) as bar  
  from histogram;
<table>
<thead>
<tr>
<th>bucket</th>
<th>range</th>
<th>freq</th>
<th>bar</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[10,15)</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[15,20)</td>
<td>1363</td>
<td>**</td>
</tr>
<tr>
<td>3</td>
<td>[20,25)</td>
<td>8832</td>
<td>*********************</td>
</tr>
<tr>
<td>4</td>
<td>[25,30)</td>
<td>20917</td>
<td>********************************************</td>
</tr>
<tr>
<td>5</td>
<td>[30,35)</td>
<td>20681</td>
<td>********************************************</td>
</tr>
<tr>
<td>6</td>
<td>[35,40)</td>
<td>9166</td>
<td>*********************</td>
</tr>
<tr>
<td>7</td>
<td>[40,45)</td>
<td>2093</td>
<td>***</td>
</tr>
<tr>
<td>8</td>
<td>[45,50)</td>
<td>247</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>[50,54)</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>[54,55)</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

(10 rows)
wCTE Queries

- WITH INSERT INTO ... RETURNING *
- Trick DAOs and ORMs
- Database Access Objects can be cool
- ORM are your enemy
with queue as (  
  insert into queue (extension)  
  select id  
  from extension  
  where shortname = $1  
  returning id, extension
)
select q.id, e.id as ext_id,  
  e.fullname, e.uri, e.description  
from queue q  
join extension e on q.extension = e.id
Advanced Indexing

# select id, name, pos, 
   round((pos <@> point(-0.12,51.516))::numeric, 3) as miles 
from pubnames 
order by pos <-> point(-0.12,51.516) 
limit 10;

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>pos</th>
<th>miles</th>
</tr>
</thead>
<tbody>
<tr>
<td>21593238</td>
<td>All Bar One</td>
<td>(-0.1192746,51.5163499)</td>
<td>0.039</td>
</tr>
<tr>
<td>26848690</td>
<td>The Shakespeare’s Head</td>
<td>(-0.1194731,51.5167871)</td>
<td>0.059</td>
</tr>
<tr>
<td>371049718</td>
<td>The Newton Arms</td>
<td>(-0.1209811,51.5163032)</td>
<td>0.047</td>
</tr>
<tr>
<td>438488621</td>
<td>Marquis Cornwallis</td>
<td>(-0.1199612,51.5146691)</td>
<td>0.092</td>
</tr>
<tr>
<td>21593236</td>
<td>Ship Tavern</td>
<td>(-0.1192378,51.5172525)</td>
<td>0.093</td>
</tr>
<tr>
<td>312156665</td>
<td>The Prince of Wales</td>
<td>(-0.121732,51.5145794)</td>
<td>0.123</td>
</tr>
<tr>
<td>312156722</td>
<td>O’Neills</td>
<td>(-0.1220195,51.5149538)</td>
<td>0.113</td>
</tr>
<tr>
<td>25508632</td>
<td>Friend at Hand</td>
<td>(-0.1224717,51.5148694)</td>
<td>0.132</td>
</tr>
<tr>
<td>338507304</td>
<td>The Square Pig</td>
<td>(-0.1191744,51.5187089)</td>
<td>0.191</td>
</tr>
<tr>
<td>1975855516</td>
<td>Holborn Whippet</td>
<td>(-0.1216925,51.5185189)</td>
<td>0.189</td>
</tr>
</tbody>
</table>
(10 rows)
Joins, Lateral Joins

```
# select c.name, 
array_to_string(array_agg(distinct(cp.name) order by cp.name), ',', ''), count(*)
from cities c,
  lateral (select name
            from pubnames p
            where (p.pos <@> c.pos) < 5
              ) as cp
where c.name <> 'Westminster'
group by c.name, replace(replace(cp.name, 'The ', ''), 'And', '&')
order by count(*) desc
limit 3;
```

<table>
<thead>
<tr>
<th>name</th>
<th>array_to_string</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>Prince of Wales, The Prince of Wales</td>
<td>15</td>
</tr>
<tr>
<td>London</td>
<td>All Bar One</td>
<td>12</td>
</tr>
<tr>
<td>London</td>
<td>The Beehive</td>
<td>8</td>
</tr>
</tbody>
</table>

jeudi 20 mars 14
JOIN in DML

WITH upd AS (  
  UPDATE target t  
    SET counter = t.counter + s.counter,  
  FROM source s  
  WHERE t.id = s.id  
  RETURNING s.id  
)  
INSERT INTO target(id, counter)  
SELECT id, sum(counter)  
FROM source s LEFT JOIN upd t USING(id)  
WHERE t.id IS NULL  
GROUP BY s.id  
RETURNING t.id
Other SQL Features

• COPY: the Streaming Protocol
• LISTEN / NOTIFY
• JSON datatype, JSON result sets
• CREATE FUNCTION
• CREATE AGGREGATE
• Functions, operators, etc
Conclusion

- Tunable ACID
- Data Types
- Functions and Operators
- Extensions
- Advanced Indexing

- Powerful SQL
- Common Table Expressions
- Writeable CTE
- Window Functions
- Aggregates
PostgreSQL is YeSQL!